Polymorphism

In Shopping malls behave like Customer

In Bus behave like Passenger

In School behave like Student

At Home behave like Son

Polymorphism

- Polymorphism is the capability of a method to do different things based on the object that it is acting upon.
- Polymorphism allows you define one interface and have multiple implementations.
- The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Types of polymorphism

- **Runtime Polymorphism(or Dynamic polymorphism)**

  Method overriding is a perfect example of runtime polymorphism.

- **Compile time Polymorphism(or Static polymorphism)**

  Method overloading is a perfect example of compile polymorphism.

Polymorphism

Method Overloading

- In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.
- Overloaded methods must have a different argument list.
- They may have the same or different return types.
- It is also known as compile time polymorphism.

Polymorphism

Method Overriding

- Child class has the same method as of base class. In such cases child class overrides the parent class method without even touching the source code of the base class. This feature is known as method overriding.
- Overriding method can have different return type
- Static and final methods cannot be overridden
- Constructors cannot be overridden
- It is also known as Runtime polymorphism.

Polymorphism

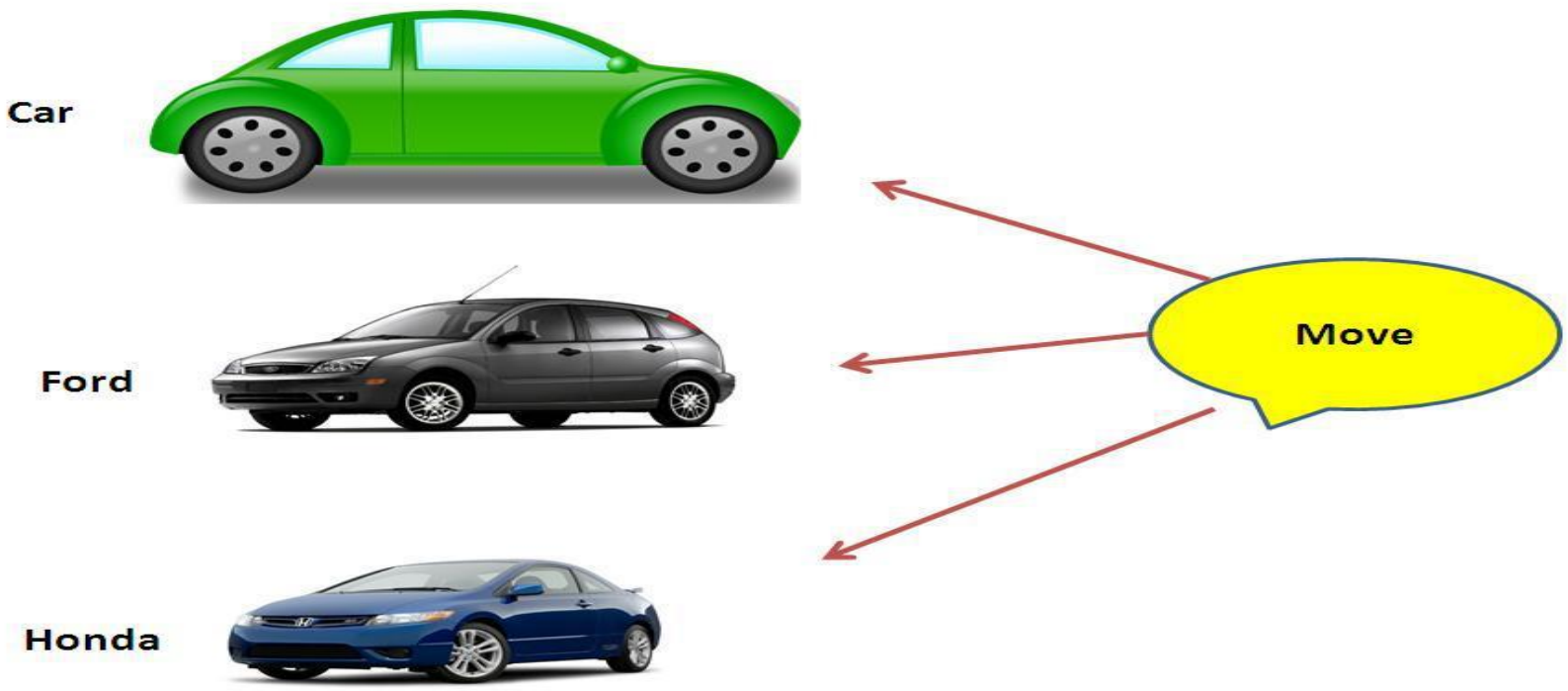
Example:

```
public interface Vegetarian{}
public class Animal{}
public class Deer extends Animal implements Vegetarian{}
```

Now, the Deer class is considered to be polymorphic since this has multiple inheritance. Following are true for the above example:

- A Deer IS-A Animal
- A Deer IS-A Vegetarian
- A Deer IS-A Deer
- A Deer IS-A Object

Polymorphism



- Car uses normal engine to move
- Ford uses V engine to move
- Honda uses i-vtec technology to move

Method Overriding

```
public class WritingInstrument{

    public void Write(){
        // Makes visible mark
    }
}


public class Pen : WritingInstrument{

    public void Write(){
        // Makes visible mark with ink
    }
}
```

Method Overloading

```
public class WritingInstrument{

    public void Write(){
        // Makes visible mark
    }

    public void Write(int fontSize){
        // Makes visible mark of given font
        // size
    }
```